```
Syntax                                                                                    word size
            rematstr ............................................................................1
            if(rematstr != reuse)
            {
                if((cplbegf > 2) || (cplinu == 0))
                {
                    for(rbnd = 0; rbnd < 4; rbnd++) {rematflg[rbnd]}.............................1
                }
                if((2 >= cplbegf > 0) && cplinu)
                {
                    for(rbnd = 0; rbnd < 3; rbnd++) {rematflg[rbnd]}.............................1
                }
                if((cplbegf == 0) && cplinu)
                {
                    for(rbnd = 0; rbnd < 2; rbnd++) {rematflg[rbnd]}.............................1
                }
            }
        }

/* These fields for exponent strategy */
        if(cplinu) {cplexpstr}...................................................................2
        for(ch = 0; ch < nfchans; ch++) {chexpstr[ch]}...........................................2
        if(lfeon) {lfeexpstr}....................................................................1
        for(ch = 0; ch < nfchans; ch++)
        {
            if(chexpstr[ch] != reuse)
            {
                if(!chincpl[ch]) {chbwcod[ch]} ..................................................6
            }
        }

/* These fields for exponents */
        if(cplinu) /* exponents for the coupling channel */
        {
            if(cplexpstr != reuse)
            {
                cplabsexp........................................................................4
                /* ncplgrps derived from ncplsubnd, cplexpstr */
                for(grp = 0; grp< ncplgrps; grp++) {cplexps[grp]} ...............................7
            }
        }
        for(ch = 0; ch < nfchans; ch++) /* exponents for full bandwidth channels */
        {
            if(chexpstr[ch] != reuse)
            {
                exps[ch][0] .....................................................................4
                /* nchgrps derived from chexpstr[ch], and cplbegf or chbwcod[ch] */
                for(grp = 1; grp <= nchgrps[ch]; grp++) {exps[ch][grp]}..........................7
                gainrng[ch] .....................................................................2
            }
        }
        if(lfeon) /* exponents for the low frequency effects channel */
        {
            if(lfeexpstr != reuse)
            {
```

```
Syntax                                                                  word size
            lfeexps[0] .................................................. 4
            /* nlfegrps = 2 */
            for(grp = 1; grp <= nlfegrps; grp++) {lfeexps[grp]} .......... 7
        }
    }

/* These fields for bit-allocation parametric information */
    baie ................................................................ 1
    if(baie)
    {
        sdcycod ......................................................... 2
        fdcycod ......................................................... 2
        sgaincod ........................................................ 2
        dbpbcod ......................................................... 2
        floorcod ........................................................ 3
    }
    snroffste ........................................................... 1
    if(snroffste)
    {
        csnroffst ....................................................... 6
        if(cplinu)
        {
            cplfsnroffst ................................................ 4
            cplfgaincod ................................................. 3
        }
        for(ch = 0; ch < nfchans; ch++)
        {
            fsnroffst[ch] ............................................... 4
            fgaincod[ch] ................................................ 3
        }
        if(lfeon)
        {
            lfefsnroffst ................................................ 4
            lfefgaincod ................................................. 3
        }
    }
    if(cplinu)
    {
        cplleake ........................................................ 1
        if(cplleake)
        {
            cplfleak .................................................... 3
            cplsleak .................................................... 3
        }
    }

/* These fields for delta bit allocation information */
    deltbaie ............................................................ 1
    if(deltbaie)
    {
        if(cplinu) {cpldeltbae} ......................................... 2
        for(ch = 0; ch < nfchans; ch++) {deltbae[ch]} ................... 2
        if(cplinu)
```

```
Syntax                                                                                word size
        {
            if(cpldeltbae==new info follows)
            {
                cpldeltnseg ..................................................................3
                for(seg = 0; seg <= cpldeltnseg; seg++)
                {
                    cpldeltoffst[seg] ........................................................5
                    cpldeltlen[seg] ..........................................................4
                    cpldeltba[seg] ...........................................................3
                }
            }
        }
        for(ch = 0; ch < nfchans; ch++)
        {
            if(deltbae[ch]==new info follows)
            {
                deltnseg[ch] .................................................................3
                for(seg = 0; seg <= deltnseg[ch]; seg++)
                {
                    deltoffst[ch][seg] .......................................................5
                    deltlen[ch][seg] .........................................................4
                    deltba[ch][seg] ..........................................................3
                }
            }
        }
    }

/* These fields for inclusion of unused dummy data */
    skiple ...........................................................................1
    if(skiple)
    {
        skipl ........................................................................9
        skipfld ..............................................................skipl × 8
    }

/* These fields for quantized mantissa values */
    ch = 0
    do /* mantissas of chs up to and including first coupled ch */
    {
        for(bin = 0; bin < nchmant[ch]; bin++) {chmant[ch][bin]} ................ (0-16)
        ch += 1
    }
    while(chinclp[ch] = = 0 && ch < nfchans)
    if(cplinu) /* mantissas of coupling channel */
    {
        for(bin = 0; bin < ncplmant; bin++) {cplmant[bin]} ...................... (0-16)
    }
    while(ch<nfchans) /* mantissas of remaining channels, whether or not coupled */
    {
        for(bin = 0; bin < nchmant[ch]; bin++) {chmant[ch][bin]} ................ (0-16)
        ch += 1
    }
    if(lfeon) /* mantissas of low frequency effects channel */
    {
```

Syntax	word size
for(bin = 0; bin < nlfemant; bin++) **{lfemant[bin]}**	(0-16)
}	
} /* end of audblk */	

5.3.4 auxdata - auxiliary data

Syntax	word size
auxdata()	
{	
auxbits	nauxbits
if(auxdatae)	
{	
auxdatal	14
}	
auxdatae	1
} /* end of auxdata */	

5.3.5 errorcheck - error detection code

Syntax	word size
errorcheck()	
{	
crcrsv	1
crc2	16
} /* end of errorcheck */	

5.4 Description Of Bit Stream Elements

A number of bit stream elements have values which may be transmitted, but whose meaning has been reserved. If a decoder receives a bit stream which contains reserved values, the decoder may or may not be able to decode and produce audio. In the description of bit stream elements which have reserved codes, there is an indication of what the decoder can do if the reserved code is received. In some cases, the decoder can not decode audio. In other cases, the decoder can still decode audio by using a default value for a parameter which was indicated by a reserved code.

5.4.1 syncinfo - Synchronization Information

5.4.1.1 syncword - Synchronization Word - 16 bits

The syncword is always 0x0B77, or 0000 1011 0111 0111. Transmission of the syncword, like other bit field elements, is left bit first.

5.4.1.2 crc1 - Cyclic Redundancy Check 1 - 16 bits

This 16 bit-CRC applies to the first 5/8 of the frame. Transmission of the CRC, like other numerical values, is most significant bit first.

5.4.1.3 fscod - Sample Rate Code - 2 bits

This is a 2-bit code indicating sample rate according to Table 5.1. If the reserved code is indicated, the decoder should not attempt to decode audio and should mute.

Table 5.1 Sample rate codes.

fscod	sampling rate, kHz
00	48
01	44.1
10	32
11	reserved

5.4.1.4 frmsizecod - Frame Size Code - 6 bits

The frame size code is used along with the sample rate code to determine the number of (2-byte) words before the next syncword. See Table 5.13 on page 37.

5.4.2 bsi - Bit Stream Information

5.4.2.1 bsid - Bit Stream Identification - 5 bits

This bit field has a value of 01000 (=8) in this version of this standard. Future modifications of this standard may define other values. Values of bsid smaller than 8 will be used for versions of AC-3 which implement subsets of the version 8 syntax. Decoders which can decode version 8 will thus be able to decode version numbers less than 8. If this standard is extended by the addition of additional elements or features, a value of bsid greater than 8 will be used. Decoders built to this version of the standard will not be able to decode versions with bsid greater than 8. Thus, decoders built to this standard shall mute if the value of bsid is greater than 8, and should decode and reproduce audio if the value of bsid is less than or equal to 8.

5.4.2.2 bsmod - Bit Stream Mode - 3 bits

This 3-bit code indicates the type of service that the bit stream conveys as defined in Table 5.2.

Table 5.2 Bit stream mode.

bsmod	type of service
000	main audio service
001	main audio service minus dialog
010	associated service: visually impaired
011	associated service: hearing impaired
100	associated service: dialog
101	associated service: commentary
110	associated service: emergency - flash
111	associated service: voice over

5.4.2.3 acmod - Audio Coding Mode - 3 bits

This 3-bit code, shown in Table 5.3, indicates which of the main service channels are in use, ranging from 3/2 to 1/0. If the msb of acmod is a 1, surround channels are in use and surmixlev follows in the bit stream. If the msb of acmod is a 0, the surround channels are not in use and surmixlev does not follow in the bit stream. If the lsb of acmod is a 0, the center channel is not in use. If the lsb of acmod is a 1, the center channel is in use. Note: The state of acmod sets the number of full-bandwidth channels parameter, nfchans, (e.g., for 3/2 mode, nfchans = 5; for 2/1 mode, nfchans = 3; etc.). The total number of channels, nchans, is equal to nfchans if the lfe channel is off, and is equal to 1+nfchans if the lfe channel is on. If acmod is 0, then two completely independent program channels (dual mono) are encoded into the bit stream, and are referenced as Ch1, Ch2. In this case, a number of additional items are present in BSI or audblk to fully describe Ch2. Table 5.3 also indicates the channel ordering (the order in which the channels are processed) for each of the modes.

Table 5.3 Audio coding mode.

acmod	audio coding mode	nfchans	channel array ordering
000	1+1	2	Ch1, Ch2
001	1/0	1	C
010	2/0	2	L, R
011	3/0	3	L, C, R
100	2/1	3	L, R, S
101	3/1	4	L, C, R, S
110	2/2	4	L, R, SL, SR
111	3/2	5	L, C, R, SL, SR

5.4.2.4 cmixlev - Center Mix Level - 2 bits

When three front channels are in use, this 2-bit code, shown in Table 5.4, indicates the nominal down mix level of the center channel with respect to the left and right channels. If cmixlev is set to the reserved code, decoders should still reproduce audio. The intermediate value of cmixlev (-4.5 dB) may be used in this case.

Table 5.4 Center mix level.

cmixlev	mix level
00	0.707 (-3.0 dB)
01	0.596 (-4.5 dB)
10	0.500 (-6.0 dB)
11	reserved

5.4.2.5 surmixlev - Surround Mix Level - 2 bits

If surround channels are in use, this 2-bit code, shown in Table 5.5, indicates the nominal down mix level of the surround channels. If surmixlev is set to the reserved code, the decoder should still reproduce audio. The intermediate value of surmixlev (-6 dB) may be used in this case.

Table 5.5 Surround mix level.

surmixlev	mix level
00	0.707 (-3 dB)
01	0.500 (-6 dB)
10	0
11	reserved

5.4.2.6 dsurmod - Dolby Surround Mode - 2 bits

When operating in the two channel mode, this 2-bit code, as shown in Table 5.6, indicates whether or not the program has been encoded in Dolby Surround. This information is not used by the AC-3 decoder, but may be used by other portions of the audio reproduction equipment. If dsurmod is set to the reserved code, the decoder should still reproduce audio. The reserved code may be interpreted as "not indicated".

Table 5.6 Dolby Surround mode.

dsurmod	indication
00	not indicated
01	NOT Dolby Surround encoded
10	Dolby Surround encoded
11	reserved

5.4.2.7 lfeon - Low Frequency Effects Channel On - 1 bit

This bit has a value of 1 if the lfe (sub woofer) channel is on, and a value of 0 if the lfe channel is off.

5.4.2.8 dialnorm - Dialogue Normalization - 5 bits

This 5-bit code indicates how far the average dialogue level is below digital 100%. Valid values are 1-31. The value of 0 is reserved. The values of 1 to 31 are interpreted as -1 dB to -31 dB with respect to digital 100%. If the reserved value of 0 is received, the decoder shall use -31 dB. The value of dialnorm shall affect the sound reproduction level. If the value is not used by the AC-3 decoder itself, the value shall be used by other parts of the audio reproduction equipment. Dialog normalization is further explained in Section 7.6 on page 73.

5.4.2.9 compre - Compression Gain Word Exists - 1 bit

If this bit is a 1, the following 8 bits represent a compression control word.

5.4.2.10 compr - Compression Gain Word - 8 bits

This encoder generated gain word may be present in the bit stream. If so, it may used to scale the reproduced audio level in order to reproduce a very narrow dynamic range, with an assured upper limit of instantaneous peak reproduced signal level in the monophonic downmix. The meaning and use of **compr** is described further in Section 7.7.2 on page 78.

5.4.2.11 langcode - Language Code Exists - 1 bit

If this bit is a 1, the following 8 bits represent a language code. If this bit is a 0, the language of the audio service is not indicated.

5.4.2.12 langcod - Language Code - 8 bits

This is an 8 bit code representing the language of the audio service. See Table 5.14 on page 38 for the mapping of langcod into language.

5.4.2.13 audprodie - Audio Production Information Exists - 1 bit

If this bit is a 1, the mixlevel and roomtyp fields exist, indicating information about the audio production environment (mixing room).

5.4.2.14 mixlevel - Mixing Level - 5 bits

This 5-bit code indicates the acoustic sound pressure level of the dialogue level during the final audio mixing session. The 5-bit code represents decibels from 0 dB to 31 dB. The reference mixing level is 60 dB SPL plus the value of mixlevel, or 60 dB to 91 dB SPL. The value of mixlevel is not typically used within the AC-3 decoder, but may be used by other parts of the audio reproduction equipment.

5.4.2.15 roomtyp - Room Type - 2 bits

This 2-bit code, shown in Table 5.7, indicates the type and calibration of the mixing room used for the final audio mixing session. The value of roomtyp is not typically used by the AC-3 decoder, but may be used by other parts of the audio reproduction equipment. If roomtyp is set to the reserved code, the decoder should still reproduce audio. The reserved code may be interpreted as "not indicated".

Table 5.7 Room type.

roomtyp	type of mixing room
00	not indicated
01	large room, X curve monitor
10	small room, flat monitor
11	reserved

5.4.2.16 dialnorm2 - Dialogue Normalization, Ch2 - 5 bits

This 5-bit code has the same meaning as dialnorm, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

5.4.2.17 compr2e - Compression Gain Word Exists, Ch2 - 1 bit

If this bit is a 1, the following 8 bits represent a compression gain word for Ch2.

5.4.2.18 compr2 - Compression Gain Word, Ch2 - 8 bits

This 8-bit word has the same meaning as compr, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

5.4.2.19 langcod2e - Language Code Exists, Ch2 - 1 bit

If this bit is a 1, the following 8 bits represent a language code for Ch2. If this bit is a 0, the language of the Ch2 is not indicated.

5.4.2.20 langcod2 - Language Code, Ch2 - 8 bits

This 8-bit code has the same meaning as langcod, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

5.4.2.21 audprodi2e - Audio Production Information Exists, Ch2 - 1 bit

If this bit is a 1, the following two data fields exist indicating information about the audio production for Ch2.

5.4.2.22 mixlevel2 - Mixing Level, Ch2 - 5 bits

This 5-bit code has the same meaning as mixlevel, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

5.4.2.23 roomtyp2 - Room Type, Ch2 - 2 bits

This 2 bit code has the same meaning as roomtyp, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

5.4.2.24 copyrightb - Copyright bit - 1 bit

If this bit has a value of 1, the information in the bit stream is indicated as protected by copyright. It has a value of 0 if the information is not indicated as protected.

5.4.2.25 origbs - Original Bitstream - 1 bit

This bit has a value of 1 if this is an original bit stream. This bit has a value of 0 if this is a copy of another bit stream.

5.4.2.26 timecod1e, timcode2e - Time Code (First and Second) Halves Exists - 2 bits

These values indicate, as shown in Table 5.8, whether time codes follow in the bit stream. The time code can have a resolution of 1/64th of a frame (one frame = 1/30th of a second). Since only the high resolution portion of the time code is needed for fine synchronization, the 28 bit time code is broken into two 14 bit halves. The low resolution first half represents the code in 8 second increments up to 24 hrs. The high resolution second half represents the code in 1/64th frame increments up to 8 seconds.

Table 5.8 Time code exists.

timecod2e,timecod1e	time code present
0,0	not present
0,1	first half (14 bits) present
1,0	second half (14 bits) present
1,1	both halves (28 bits) present

5.4.2.27 timecod1 - Time Code First Half - 14 bits

The first 5 bits of this 14 bit field represent the time in hours, with valid values of 0-23. The next 6 bits represent the time in minutes, with valid values of 0-59. The final 3 bits represents the time in 8 second increments, with valid values of 0-7 (representing 0, 8, 16, ... 56 seconds).

5.4.2.28 timecod2 - Time Code Second Half - 14 bits

The first 3 bits of this 14 bit field represent the time in seconds, with valid values from 0-7 (representing 0-7 seconds). The next 5 bits represents the time in frames, with valid values from 0-29. The final 6 bits represents fractions of 1/64 of a frame, with valid values from 0-63.

5.4.2.29 addbsie - Additional Bitstream Information Exists - 1 bit

If this bit has a value of 1 there is additional bit stream information, the length of which is indicated by the next field. If this bit has a value of 0, there is no additional bit stream information.

5.4.2.30 addbsil - Additional Bitstream Information Length - 6 bits

This 6-bit code, which exists only if addbside is a 1, indicates the length in bytes of additional bit stream information. The valid range of addbsil is 0-63, indicating 1-64 additional bytes, respectively. The decoder is not required to interpret this information, and thus shall skip over this number of bytes following in the data stream.

5.4.2.31 addbsi - Additional Bitstream Information - ((addbsil+1) $\times$ 8) bits

This field contains 1 to 64 bytes of any additional information included with the bit stream information structure.

5.4.3 audblk - Audio Block

5.4.3.1 blksw[ch] - Block Switch Flag - 1 bit

This flag, for channel [ch], indicates whether the current audio block was split into 2 sub-blocks during the transformation from the time domain into the frequency domain. A value of 0 indicates that the block was not split, and that a single 512 point TDAC transform was performed. A value of 1 indicates that the block was split into 2 sub-blocks of length 256, that the TDAC transform length was switched from a length of 512 points to a length of 256 points, and that 2 transforms were performed on the audio block (one on each sub-block). Transform length switching is described in more detail in Section 7.9 on page 85.

5.4.3.2 dithflag[ch] - Dither Flag - 1 bit

This flag, for channel [ch], indicates that the decoder should activate dither during the current block. Dither is described in detail in Section 7.3.4 on page 66.

5.4.3.3 dynrnge - Dynamic Range Gain Word Exists - 1 bit

If this bit is a 1, the dynamic range gain word follows in the bit stream. If it is 0, the gain word is not present, and the previous value is reused, except for block 0 of a frame where if the control word is not present the current value of dynrng is set to 0.

5.4.3.4 dynrng - Dynamic Range Gain Word - 8 bits

This encoder-generated gain word is applied to scale the reproduced audio as described in Section 7.7.1 on page 75.

5.4.3.5 dynrng2e - Dynamic Range Gain Word Exists, Ch2 - 1 bit

If this bit is a 1, the dynamic range gain word for channel 2 follows in the bit stream. If it is 0, the gain word is not present, and the previous value is reused, except for block 0 of a frame where if the control word is not present the current value of dynrng2 is set to 0.

5.4.3.6 dynrng2 - Dynamic Range Gain Word, Ch2 - 8 bits

This encoder-generated gain word is applied to scale the reproduced audio of Ch2, in the same manner as dynrng is applied to Ch1, as described in Section 7.7.1 on page 75.

5.4.3.7 cplstre - Coupling Strategy Exists - 1 bit

If this bit is a 1, coupling information follows in the bit stream. If it is 0, new coupling information is not present, and coupling parameters previously sent are reused.

5.4.3.8 cplinu - Coupling In Use - 1 bit

If this bit is a 1, coupling is currently being utilized, and coupling parameters follow. If it is 0, coupling is not being utilized (all channels are independent) and no coupling parameters follow in the bit stream.

5.4.3.9 chincpl[ch] - Channel In Coupling - 1 bit

If this bit is a 1, then the channel indicated by the index [ch] is a coupled channel. If the bit is a 0, then this channel is not coupled. Since coupling is not used in the 1/0 mode, if any chincpl[] values exist there will be 2 to 5 values. Of the values present, at least two values will be 1, since coupling requires more than one coupled channel to be coupled.

5.4.3.10 phsflginu - Phase Flags In Use - 1 bit

If this bit (defined for 2/0 mode only) is a 1, phase flags are included with coupling coordinate information. Phase flags are described in Section 7.4 on page 68.

5.4.3.11 cplbegf - Coupling Begin Frequency Code - 4 bits

This 4-bit code is interpreted as the subband number (0 to 15) which indicates the lower frequency band edge of the coupling channel (or the first active subband) as shown in Table 7.24 on page 69.

5.4.3.12 cplendf - Coupling End Frequency Code - 4 bits

This 4-bit code indicates the upper band edge of the coupling channel. The upper band edge (or last active subband) is cplendf+2, or a value between 2 and 17. See Table 7.24 on page 69.

The number of active coupling subbands is equal to ncplsubnd, which is calculated:

```
ncplsubnd = 3 + cplendf - cplbegf ;
```

5.4.3.13 cplbndstrc[sbnd] - Coupling Band Structure - 1 bit

There are 18 coupling subbands defined in Table 7.24 on page 69, each containing 12 frequency coefficients. The fixed 12-bin wide coupling subbands are converted into coupling bands, each of which may be wider than (a multiple of) 12 frequency bins. Each coupling band may contain one or more coupling subbands. Coupling coordinates are transmitted for each coupling band. Each band's coupling coordinate must be applied to all the coefficients in the coupling band.

The coupling band structure indicates which coupling subbands are combined into wider coupling bands. When cplbndstrc[sbnd] is a 0, the subband number [sbnd] is not combined into the previous band to form a wider band, but starts a new 12 wide coupling band. When cplbndstrc[sbnd] is a 1, then the subband [sbnd] is combined with the previous band, making the previous band 12 bins wider. Each successive value of cplbndstrc which is a 1 will continue to combine subbands into the current band. When another cplbndstrc value of 0 is received, then a new band will be formed, beginning with the 12 bins of the current subband. The set of cplbndstrc[sbnd] values is typically considered an array.

Each bit in the array corresponds to a specific coupling subband in ascending frequency order. The first element of the array corresponds to the subband cplbegf, is always 0, and is not transmitted. (There is no reason to send a cplbndstrc bit for the first subband at cplbegf, since this bit would always be 0.) Thus, there are ncplsubnd-1 values of cplbndstrc transmitted. If there is only one coupling subband, then no cplbndstrc bits are sent.

The number of coupling bands, ncplbnd, may be computed from ncplsubnd and cplbnstrc:

```
ncplbnd = (ncplsubnd - (cplbndstrc[cplbegf+1] + ... + cplbndstrc[cplendf+2])) ;
```

5.4.3.14 cplcoe[ch] - Coupling Coordinates Exist - 1 bit

Coupling coordinates indicate, for a given channel and within a given coupling band, the fraction of the coupling channel frequency coefficients to use to re-create the individual channel frequency coefficients. Coupling coordinates are conditionally transmitted in the bit stream. If new values are not delivered, the previously sent values remain in effect. See Section 7.4 on page 68 for further information on coupling.

If cplcoe[ch] is 1, the coupling coordinates for the corresponding channel [ch] exist and follow in the bit stream. If the bit is 0, the previously transmitted coupling coordinates for this channel are reused. All coupling coordinates are always transmitted in block 0 of each syncframe.

5.4.3.15 mstrcplco[ch] - Master Coupling Coordinate - 2 bits

This per channel parameter establishes a per channel gain factor (increasing the dynamic range) for the coupling coordinates as shown in Table 5.9.

Table 5.9 Master coupling coordinate.

mstrcplco[ch]	cplco[ch][bnd] gain multiplier
00	1
01	2^{-3}
10	2^{-6}
11	2^{-9}

5.4.3.16 cplcoexp[ch][bnd] - Coupling Coordinate Exponent - 4 bits

Each coupling coordinate is composed of a 4-bit exponent and a 4-bit mantissa. This element is the value of the coupling coordinate exponent for channel [ch] and band [bnd]. The index [ch] only will exist for those channels which are coupled. The index [bnd] will range from 0 to ncplbnds. See Section 7.4.3 on page 70 for further information on how to interpret coupling coordinates.

5.4.3.17 cplcomant[ch][bnd] - Coupling Coordinate Mantissa- 4 bits

This element is the 4-bit coupling coordinate mantissa for channel [ch] and band [bnd].

5.4.3.18 phsflg[bnd] - Phase Flag - 1 bit

This element (only used in the 2/0 mode) indicates whether the decoder should phase invert the coupling channel mantissas when reconstructing the right output channel. The index [bnd] can range from 0 to ncplbnd. Phase flags are described in Section 7.4 on page 68.

5.4.3.19 rematstr - Rematrixing Strategy - 1 bit

If this bit is a 1, then new rematrix flags are present in the bit stream. If it is 0, rematrix flags are not present, and the previous values should be reused. The rematstr parameter is present only in the 2/0 audio coding mode.

5.4.3.20 rematflg[rbnd] - Rematrix Flag - 1 bit

This bit indicates whether the transform coefficients in rematrixing band [rbnd] have been rematrixed. If this bit is a 1, then the transform coefficients in [rbnd] were rematrixed into sum and difference channels. If this bit is a 0, then rematrixing has not been performed in band [rbnd]. The number of rematrixing bands (and the number of values of [rbnd]) depend on coupling parameters as shown in Table 5.10. Rematrixing is described in Section 7.5 on page 70.

Table 5.10 Number of rematrixing bands.

condition	# of rematrixing bands
cplinu == 0	4
(cplinu == 1) && (cplbegf > 2)	4
(cplinu == 1) && (2 ≥ cplbegf > 0)	3
(cplinu == 1) && (cplbegf == 0)	2

5.4.3.21 cplexpstr - Coupling Exponent Strategy - 2 bits

This element indicates the method of exponent coding that is used for the coupling channel as shown in Table 7.24 on page 69. See Section 7.1 on page 44 for explanation of each exponent strategy.

5.4.3.22 chexpstr[ch] - Channel Exponent Strategy - 2 bits

This element indicates the method of exponent coding that is used for channel [ch], as shown in Table 7.24 on page 69. This element exists for each full bandwidth channel.

5.4.3.23 lfeexpstr - Low Frequency Effects Channel Exponent Strategy - 1 bit

This element indicates the method of exponent coding that is used for the lfe channel, as shown in Table 7.24 on page 69.

5.4.3.24 chbwcod[ch] - Channel Bandwidth Code - 6 bits

The chbwcod[ch] element is an unsigned integer which defines the upper band edge for full-bandwidth channel [ch]. This parameter is only included for fbw channels which are not coupled. (See Section 7.1.3 on page 46 on exponents for the definition of this parameter.) Valid values are in the range of 0-60. If a value greater than 60 is received, the bit stream is invalid and the decoder shall cease decoding audio and mute.

5.4.3.25 cplabsexp - Coupling Absolute Exponent - 4 bits

This is an absolute exponent, which is used as a reference when decoding the differential exponents for the coupling channel.

5.4.3.26 cplexps[grp] - Coupling Exponents - 7 bits

Each value of cplexps indicates the value of 3, 6, or 12 differentially-coded coupling channel exponents for the coupling exponent group [grp] for the case of D15, D25, or D45 coding, respectively. The number of cplexps values transmitted equals ncplgrps, which may be determined from cplbegf, cplendf, and cplexpstr. Refer to Section 7.1.3 on page 46 for further information.

5.4.3.27 exps[ch][grp] - Channel Exponents - 4 or 7 bits

These elements represent the encoded exponents for channel [ch]. The first element ([grp]=0) is a 4-bit absolute exponent for the first (DC term) transform coefficient. The subsequent elements ([grp]>0) are 7-bit representations of a group of 3, 6, or 12 differentially coded exponents (corresponding to D15, D25, D45 exponent strategies respectively). The number of groups for each channel, nchgrps[ch], is determined from cplbegf if the channel is coupled, or chbwcod[ch] of the channel is not coupled. Refer to Section 7.1.3 on page 46 for further information.

5.4.3.28 gainrng[ch] - Channel Gain Range Code - 2 bits

This per channel 2-bit element may be used to determine a block floating-point shift value for the inverse TDAC transform filterbank. Use of this code allows increased dynamic range to be obtained from a limited word length transform computation. For further information see Section 7.9 on page 85.

5.4.3.29 lfeexps[grp] - Low Frequency Effects Channel Exponents - 4 or 7 bits

These elements represent the encoded exponents for the LFE channel. The first element ([grp]=0) is a 4-bit absolute exponent for the first (DC term) transform coefficient. There are two additional elements (nlfegrps=2) which are 7-bit representations of a group of 3 differentially coded exponents. The total number of lfe channel exponents (nlfemant) is 7.

5.4.3.30 baie - Bit Allocation Information Exists - 1 bit

If this bit is a 1, then five separate fields (totaling 11 bits) follow in the bit stream. Each field indicates parameter values for the bit allocation process. If this bit is a 0, these fields do not exist. Further details on these fields may be found in Section 7.2 on page 49.

5.4.3.31 sdcycod - Slow Decay Code - 2 bits

This is a 2-bit code specifies the slow decay parameter in the bit allocation process.

5.4.3.32 fdcycod - Fast Decay Code - 2 bits

This is a 2-bit code specifies the fast decay parameter in the decode bit allocation process.

5.4.3.33 sgaincod - Slow Gain Code - 2 bits

This is a 2-bit code specifies the slow gain parameter in the decode bit allocation process.

5.4.3.34 dbpbcod - dB Per Bit Code - 2 bits

This 2-bit code specifies the dB per bit parameter in the bit allocation process.

5.4.3.35 floorcod - Masking Floor Code - 3 bits

This 3-bit code specifies the floor code parameter in the bit allocation process.

5.4.3.36 snroffste - SNR Offset Exists - 1 bit

If this bit has a value of 1, a number of bit allocation parameters follow in the bit stream. If this bit has a value of 0, SNR offset information does not follow, and the previously transmitted values should be used for this block. The bit allocation process and these parameters are described in Section 7.2.2 on page 50.

5.4.3.37 csnroffst - Coarse SNR Offset - 6 bits

This 6-bit code specifies the coarse snr offset parameter in the bit allocation process.

5.4.3.38 cplfsnroffst - Coupling Fine SNR Offset - 4 bits

This 4-bit code specifies the coupling channel fine snr offset in the bit allocation process.

5.4.3.39 cplfgaincod - Coupling Fast Gain Code - 3 bits

This 3-bit code specifies the coupling channel fast gain code used in the bit allocation process.

5.4.3.40 fsnroffst[ch] - Channel Fine SNR Offset - 4 bits

This 4-bit code specifies the fine snr offset used in the bit allocation process for channel [ch].

5.4.3.41 fgaincod[ch] - Channel Fast Gain Code - 3 bits

This 3-bit code specifies the fast gain parameter used in the bit allocation process for channel [ch].

5.4.3.42 lfefsnroffst - Low Frequency Effects Channel Fine SNR Offset - 4 bits

This 4-bit code specifies the fine snr offset parameter used in the bit allocation process for the lfe channel.

5.4.3.43 lfefgaincod - Low Frequency Effects Channel Fast Gain Code - 3 bits

This 3-bit code specifies the fast gain parameter used in the bit allocation process for the lfe channel.

5.4.3.44 cplleake - Coupling Leak Initialization Exists - 1 bit

If this bit is a 1, leak initialization parameters follow in the bit stream. If this bit is a 0, the previously transmitted values still apply.

5.4.3.45 cplfleak - Coupling Fast Leak Initialization - 3 bits

This 3-bit code specifies the fast leak initialization value for the coupling channel's excitation function calculation in the bit allocation process.

5.4.3.46 cplsleak - Coupling Slow Leak Initialization - 3 bits

This 3-bit code specifies the slow leak initialization value for the coupling channel's excitation function calculation in the bit allocation process.

5.4.3.47 deltbaie - Delta Bit Allocation Information Exists - 1 bit

If this bit is a 1, some delta bit allocation information follows in the bit stream. If this bit is a 0, the previously transmitted delta bit allocation information still applies, except for block 0. If deltbaie is 0 in block 0, then cpldeltnseg and deltnseg[ch] are set to 0, and no delta bit allocation is applied. Delta bit allocation is described in Section 7.2.2.6 on page 55.

5.4.3.48 cpldeltbae - Coupling Delta Bit Allocation Exists - 2 bits

This 2-bit code indicates the delta bit allocation strategy for the coupling channel, as shown in Table 5.11. If the reserved state is received, the decoder should not decode audio, and should mute.

Table 5.11 Delta bit allocation exist states.

cpldeltbae, deltbae	code
00	reuse previous state
01	new info follows
10	perform no delta alloc
11	reserved

5.4.3.49 deltbae[ch] - Delta Bit Allocation Exists - 2 bits

This per full bandwidth channel 2-bit code indicates the delta bit allocation strategy for the corresponding channel, as shown in Table 5.11.

5.4.3.50 cpldeltnseg - Coupling Delta Bit Allocation Number of Segments - 3 bits

This 3-bit code indicates the number of delta bit allocation segments that exist for the coupling channel. The value of this parameter ranges from 1 to 8, and is calculated by adding 1 to the 3-bit binary number represented by the code.

5.4.3.51 cpldeltoffst[seg] - Coupling Delta Bit Allocation Offset - 5 bits

The first 5-bit code ([seg]=0) indicates the number of the first bit allocation band (as specified in 7.4.2 on page 69) of the coupling channel for which delta bit allocation values are provided. Subsequent codes indicate the offset from the previous delta segment end point to the next bit allocation band for which delta bit allocation values are provided.

5.4.3.52 cpldeltlen[seg] - Coupling Delta Bit Allocation Length - 4 bits

Each 4-bit code indicates the number of bit allocation bands that the corresponding segment spans.

5.4.3.53 cpldeltba[seg] - Coupling Delta Bit Allocation - 3 bits

This 3-bit value is used in the bit allocation process for the coupling channel.

Each 3-bit code indicates an adjustment to the default masking curve computed in the decoder. The deltas are coded as shown in Table 5.12.

Table 5.12 Bit allocation deltas.

cpldeltba, deltba	adjustment
000	-24 dB
001	-18 dB
010	-12 dB
011	-6 dB
100	+6 dB
101	+12 dB
110	+18 dB
111	+24 dB

5.4.3.54 deltnseg[ch] - Channel Delta Bit Allocation Number of Segments - 3 bits

These per full bandwidth channel elements are 3-bit codes indicating the number of delta bit allocation segments that exist for the corresponding channel. The value of this parameter ranges from 1 to 8, and is calculated by adding 1 to the 3-bit binary code.

5.4.3.55 deltoffst[ch][seg] - Channel Delta Bit Allocation Offset - 5 bits

The first 5-bit code ([seg]=0) indicates the number of the first bit allocation band (see Section 7.2.2.6 on page 55) of the corresponding channel for which delta bit allocation values are provided. Subsequent codes indicate the offset from the previous delta segment end point to the next bit allocation band for which delta bit allocation values are provided.

5.4.3.56 deltlen[ch][seg] - Channel Delta Bit Allocation Length - 4 bits

Each 4-bit code indicates the number of bit allocation bands that the corresponding segment spans.

5.4.3.57 deltba[ch][seg] - Channel Delta Bit Allocation - 3 bits

This 3-bit value is used in the bit allocation process for the indicated channel. Each 3-bit code indicates an adjustment to the default masking curve computed in the decoder. The deltas are coded as shown in Table 5.12.

5.4.3.58 skiple - Skip Length Exists - 1 bit

If this bit is a 1, then the skipl parameter follows in the bit stream. If this bit is a 0, skipl does not exist.

5.4.3.59 skipl - Skip Length - 9 bits

This 9-bit code indicates the number of dummy bytes to skip (ignore) before unpacking the mantissas of the current audio block.

5.4.3.60 skipfld - Skip Field - (skipl × 8) bits

This field contains the null bytes of data to be skipped, as indicated by the skipl parameter.

5.4.3.61 chmant[ch][bin] - Channel Mantissas - 0 to 16 bits

The actual quantized mantissa values for the indicated channel. Each value may contain from 0 to as many as 16 bits. The number of mantissas for the indicated channel is equal to nchmant[ch], which may be determined from chbwcod[ch] (see Section 7.1.3 on page 46) if the channel is not coupled, or from cplbegf (see Section 7.4.2 on page 69) if the channel is coupled. Detailed information on packed mantissa data is in Section 7.3 on page 64.

5.4.3.62 cplmant[bin] - Coupling Mantissas - 0 to 16 bits

The actual quantized mantissa values for the coupling channel. Each value may contain from 0 to as many as 16 bits. The number of mantissas for the coupling channel is equal to ncplmant, which may be determined from:

ncplmant = 12 × ncplsubnd.

5.4.3.63 lfemant[bin] - Low Frequency Effects Channel Mantissas - 0 to 16 bits

The actual quantized mantissa values for the lfe channel. Each value may contain from 0 to as many as 16 bits. The value of nlfemant is 7, so there are 7 mantissa values for the lfe channel.

5.4.4 auxdata - Auxiliary Data Field

Unused data at the end of a frame will exist whenever the encoder does not utilize all available data for encoding the audio signal. This may occur if the final bit allocation falls short of using all available bits, or if the input audio signal simply does not require all available bits to be coded transparently. Or, the encoder may be instructed to intentionally leave some bits unused by audio so that they are available for use by auxiliary data. Since the number of bits required for auxiliary data may be smaller than the number of bits available (which will be time varying) in any particular frame, a method is provided to signal the number of actual auxiliary data bits in each frame.

5.4.4.1 auxbits - Auxiliary Data Bits - nauxbits bits

This field contains auxiliary data. The total number of bits in this field is:

nauxbits = (bits in frame) - (bits used by all bit stream elements except for auxbits) ;

The number of bits in the frame can be determined from the frame size code (frmsizcod) and Table 5.13 on page 37. The number of bits used includes all bits used by bit stream elements with the exception of auxbits. Any dummy data which has been included with skip fields (skipfld) is included in the used bit count. The length of the auxbits field is adjusted by the encoder such that the crc2 element falls on the last 16-bit word of the frame.

If the number of user bits indicated by auxdatal is smaller than the number of available aux bits nauxbits, the user data is located at the end of the auxbits field. This allows a decoder to find and unpack the auxdatal user bits without knowing the value of nauxbits (which can only be determined by decoding the audio in the entire frame). The order of the user data in the auxbits field is forward. Thus the aux data decoder (which may not decode any audio) may simply look to the end of the AC-3 syncframe to find auxdatal, backup auxdatal bits (from the beginning of auxdatal) in the data stream, and then unpack auxdatal bits moving forward in the data stream.

5.4.4.2 auxdatal - Auxiliary Data Length - 14 bits

This 14-bit integer value indicates the length, in bits, of the user data in the auxbits auxiliary field.

5.4.4.3 auxdatae - Auxiliary Data Exists - 1 bit

If this bit is a 1, then the auxdatal parameter precedes in the bit stream. If this bit is a 0, auxdatal does not exist, and there is no user data.

5.4.5 errorcheck - Frame Error Detection Field

5.4.5.1 crcrsv - CRC Reserved Bit - 1 bit

Reserved for use in specific applications to ensure crc2 will not be equal to the sync word. Use of this bit is optional by encoders. If the crc2 calculation results in a value equal to the syncword, the crcrsv bit may be inverted. This will result in a crc2 value which is not equal to the syncword.

5.4.5.2 crc2 - Cyclic Redundancy Check 2 - 16 bits

The 16 bit CRC applies to the entire frame. The details of the CRC checking are described in Section 7.10.1 on page 92.

5.5 Bit Stream Constraints

The following constraints are placed upon the encoded bit stream by the AC-3 encoder. These constraints allow AC-3 decoders to be manufactured with smaller input memory buffers.

1. The size of block 0 and block 1 combined, will never exceed 5/8 of the frame.
2. The sum of block 5 mantissa data and auxiliary data will never exceed the final 3/8 of the frame.
3. Block 0 always contains all necessary information to begin correctly decoding the bit stream.
4. Whenever the state of cplinu changes from off to on, all coupling information is included in the block in which coupling is turned on. No coupling related information is reused from any previous blocks where coupling may have been on.

Table 5.13 Frame size code table.

frmsizecod	nominal bitrate	fs = 32 kHz words/syncframe	fs = 44.1 kHz words/syncframe	fs = 48 kHz words/syncframe
000000 (0)	32 kb/s	96	69	64
000001 (0)	32 kb/s	96	70	64
000010 (1)	40 kb/s	120	87	80
000011 (1)	40 kb/s	120	88	80
000100 (2)	48 kb/s	144	104	96
000101 (2)	48 kb/s	144	105	96
000110 (3)	56 kb/s	168	121	112
000111 (3)	56 kb/s	168	122	112
001000 (4)	64 kb/s	192	139	128
001001 (4)	64 kb/s	192	140	128
001010 (5)	80 kb/s	240	174	160
001011 (5)	80 kb/s	240	175	160
001100 (6)	96 kb/s	288	208	192
001101 (6)	96 kb/s	288	209	192
001110 (7)	112 kb/s	336	243	224
001111 (7)	112 kb/s	336	244	224
010000 (8)	128 kb/s	384	278	256
010001 (8)	128 kb/s	384	279	256
010010 (9)	160 kb/s	480	348	320
010011 (9)	160 kb/s	480	349	320
010100 (10)	192 kb/s	576	417	384
010101 (10)	192 kb/s	576	418	384
010110 (11)	224 kb/s	672	487	448
010111 (11)	224 kb/s	672	488	448
011000 (12)	256 kb/s	768	557	512
011001 (12)	256 kb/s	768	558	512
011010 (13)	320 kb/s	960	696	640
011011 (13)	320 kb/s	960	697	640
011100 (14)	384 kb/s	1152	835	768
011101 (14)	384 kb/s	1152	836	768
011110 (15)	448 kb/s	1344	975	896
011111 (15)	448 kb/s	1344	976	896
100000 (16)	512 kb/s	1536	1114	1024
100001 (16)	512 kb/s	1536	1115	1024
100010 (17)	576 kb/s	1728	1253	1152
100011 (17)	576 kb/s	1728	1254	1152
100100 (18)	640 kb/s	1920	1393	1280
100101 (18)	640 kb/s	1920	1394	1280

Table 5.14 Language code table.

langcod	language	langcod	language	langcod	language	langcod	language
00	unknown/not applicable	20	Polish	40	background sound/clean feed	60	Moldavian
01	Albanian	21	Portuguese	41		61	Malaysian
02	Breton	22	Romanian	42		62	Malagasay
03	Catalan	23	Romansh	43		63	Macedonian
04	Croatian	24	Serbian	44		64	Laotian
05	Welsh	25	Slovak	45	Zulu	65	Korean
06	Czech	26	Slovene	46	Vietnamese	66	Khmer
07	Danish	27	Finnish	47	Uzbek	67	Kazakh
08	German	28	Swedish	48	Urdu	68	Kannada
09	English	29	Turkish	49	Ukrainian	69	Japanese
0A	Spanish	2A	Flemish	4A	Thai	6A	Indonesian
0B	Esperanto	2B	Walloon	4B	Telugu	6B	Hindi
0C	Estonian	2C		4C	Tatar	6C	Hebrew
0D	Basque	2D		4D	Tamil	6D	Hausa
0E	Faroese	2E		4E	Tadzhik	6E	Gurani
0F	French	2F		4F	Swahili	6F	Gujurati
10	Frisian	30	reserved for nat'l assignment	50	Sranan Tongo	70	Greek
11	Irish	31	"	51	Somali	71	Georgian
12	Gaelic	32	"	52	Sinhalese	72	Fulani
13	Galician	33	"	53	Shona	73	Dari
14	Icelandic	34	"	54	Serbo-Croat	74	Churash
15	Italian	35	"	55	Ruthenian	75	Chinese
16	Lappish	36	"	56	Russian	76	Burmese
17	Latin	37	"	57	Quechua	77	Bulgarian
18	Latvian	38	"	58	Pustu	78	Bengali
19	Luxembourgian	39	"	59	Punjabi	79	Belorussian
1A	Lithuanian	3A	"	5A	Persian	7A	Bambora
1B	Hungarian	3B	"	5B	Papamiento	7B	Azerbijani
1C	Maltese	3C	"	5C	Oriya	7C	Assamese
1D	Dutch	3D	"	5D	Nepali	7D	Armenian
1E	Norwegian	3E	"	5E	Ndebele	7E	Arabic
1F	Occitan	3F	"	5F	Marathi	7F	Amharic

6. DECODING THE AC-3 BIT STREAM

6.1 Introduction

Section 5 of this standard specifies the details of the AC-3 bit stream syntax. This section gives an overview of the AC-3 decoding process as diagrammed in Figure 6.1, where the decoding process flow is shown as a sequence of blocks down the center of the page, and some of the information flow is indicated by arrowed lines at the sides of the page. More detailed information on some of the processing blocks will be found in Section 7. The decoder described in this section should be considered one example of a decoder. Other methods may exist to implement decoders, and these other methods may have advantages in certain areas (such as instruction count, memory requirement, number of transforms required, etc.).

6.2 Summary of the Decoding Process

6.2.1 Input Bit Stream

The input bit stream will typically come from a transmission or storage system. The interface between the source of AC-3 data and the AC-3 decoder is not specified in this standard. The details of the interface effect a number of decoder implementation details.

6.2.1.1 Continuous or Burst Input

The encoded AC-3 data may be input to the decoder as a continuous data stream at the nominal bit-rate, or chunks of data may be burst into the decoder at a high rate with a low duty cycle. For burst mode operation, either the data source or the decoder may be the master controlling the burst timing. The AC-3 decoder input buffer may be smaller in size if the decoder can request bursts of data on an as-needed basis. However, the external buffer memory may be larger in this case.

6.2.1.2 Byte or Word Alignment

Most applications of this standard will convey the elementary AC-3 bit stream with byte or (16-bit) word alignment. The syncframe is always an integral number of words in length. The decoder may receive data as a continuous serial stream of bits without any alignment. Or, the data may be input to the decoder with either byte or word (16-bit) alignment. Byte or word alignment of the input data may allow some simplification of the decoder. Alignment does reduce the probability of false detection of the sync word.